(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 917 043 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
19.05.1999 Bulletin 1999/20

(51) Int Cl.[6]: **G06F 3/12**

(21) Application number: **98309098.6**

(22) Date of filing: **06.11.1998**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **07.11.1997 US 966406**

(71) Applicant: **XEROX CORPORATION**
Rochester, New York 14644 (US)

(72) Inventors:
- **Yang, Jennifer Y.**
  Manhattan Beach, California 90266 (US)
- **Troung, Ton Huu**
  Westminster, California 92683 (US)
- **Nesbitt, David P.**
  Redondo Beach, California 90278 (US)

(74) Representative: **Skone James, Robert Edmund**
GILL JENNINGS & EVERY
Broadgate House
7 Eldon Street
London EC2M 7LH (GB)

(54) Dynamic extension of print capabilities

(57) A utility that defines additional attributes that would cater to a user's needs provides dynamically extended printing capabilities. The system architecture allows the information to be pushed down transparently to the receiving end, which understands the semantics of the given information. System administrators can define information to monitor for accounting purposes, etc. The utility allows additional printer features to be incorporated without disrupting the existing system.

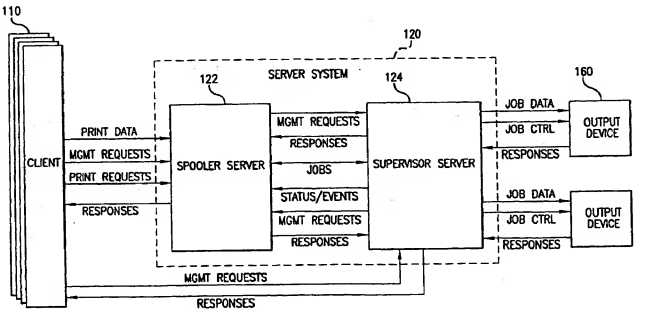

FIG.1

EP 0 917 043 A2

Description

[0001] The invention relates to dynamically extending printing capabilities using dynamic object identifiers. In particular, this invention is directed to enabling a user to define additional attributes without having to modify existing software.

[0002] The enterprise print management systems provide the means to control and access various printers and to manage other related information remotely. However, in the current systems, the extent to which these functions can be utilized is limited by the fixed set of predefined system attributes. Attributes are collections of data that describe the entities that compose the print management system. For example, printer attributes may describe the various printing features that users may use to produce high-quality documents, or they may describe status or configuration information, such as printer's state or location. Hence, when new printers. with new features not defined by existing printing standards. become available in the marketplace, the users cannot readily access these features via outdated print management system. And consequently, the print management systems vendor must upgrade the software and redistribute it to their customers.

[0003] This invention provides a method and apparatus to dynamically extend printing capabilities through the use of dynamic attributes and syntaxes. By using dynamic attributes, the system administrators may define additional attributes that would cater to their site-specific needs (e.g. accounting), and the printer vendors may plug-in the new features to the print management system without any disruption. In addition, these dynamic attributes may take values with syntax that do not adhere to the current standard.

[0004] A file format is defined to describe the new attribute, its OID (globally unique "object identifier"), and the syntax of its values. The new attributes can be read-in and registered into the system through a system administration function while the system is down, or automatically without bringing down the system. A unique algorithm is used to break-down the complex attributes into components using simple Document Printing Application (DPA) -defined syntaxes.

[0005] These and other features and advantages of this invention are described in or are apparent from the following detailed description of the preferred embodiments.

[0006] The invention is described in detail with reference to the following drawings, wherein like numerals represent like elements and wherein:

Fig. 1 is a diagram showing the interaction between the client, server and printer;
Fig. 2 is a diagram showing the flow of document data in a print request
Fig. 3 is a diagram showing the flow of attributes in detail;
Figs. 4 and 5 are diagrams showing the methods of introducing dynamic attributes to the system;
Figs. 6A and 6B is a flowchart outlining how the dynamic OIDs are processed;
Fig. 7 is an example of ASN.1 syntax for a new feature of newly released printer; and
Fig. 8 is the pseudo C structure for the example in Fig. 7.

[0007] Attributes define or characterize print management system's abstract entities, or objects. For example, document attributes, such as plex, margin, orientation, etc., describe how the printed material should appear. Printer Attributes, such as media-ready, fonts-ready, etc., describe the available resources or features of the printer. In addition to these attributes, there are a suite of attributes to facilitate end-user, operator and administrator functions. In summary, attributes are a set of data that describes the objects of the print management system.

[0008] There are three basic elements to an attribute: an object identifier (OID), a syntax and a value. An OID is a globally unique identifier of an attribute which is coded such that it may be understood by all printing systems. The OIDs are allocated following a tree format, such that each printer vendor or standards organization is assigned a branch of this tree. Then each organization may assign unique OIDs by further branching out. For example, the OID of "job-owner" is 1.0.10175.1.3.1.3 in the ISO 10175 standard for Document Printing Applications (DPA). If the server receives the "job-owner" attribute, the server can store the attribute or send this information to an account log upon completion of a print job using the OIDs, for example. Then, an accounting program written by a third party vendor can easily interpret the information in the accounting log through the OIDs.

[0009] The syntax of an attribute is the format in which the attribute value is represented. For example, the syntax of the "job-owner" attribute is "distinguishedName", which is an example of coded language the computer or printing system understands.

[0010] The value of an attribute is the instance of the attribute. For example, the value of "job-owner" could be "John Jones", i.e., the name of the person to whom the print job belongs.

[0011] Figure 1 is a diagram showing the interaction between the client 110, the server 120 and the output device 160. The output device is, for example, a printer. The client 110 is the interface between the user and the print management system. A spooler 122 takes print requests from multiple clients 110, schedules print jobs based on the print requests and then forwards the print jobs to a supervisor 124. The supervisor 124 provides the common interface between the spooler 122 and the output devices 160The supervisor 124 takes the print jobs from the spooler 122 and invokes the designated printer to render the data.

[0012] The client 110 may communicate directly with either the spooler 122 or the supervisor 124 regarding management requests. For example, the system administrator may send a request to the supervisor 124 to modify the access control information, or an end-user may send a request to the supervisor 124 to see the current status of all the printers that the supervisor 124 manages. These print requests and management requests basically communicate the characteristics of the objects of interest. In other words, the information that flows between the client 110 and the server 120 are the attributes and the corresponding values of the objects. For instance, in a print request, the client 110 sends the document data along with document attributes and corresponding values that dictate how to print, what printer to use, who the job owner is, etc. For management requests, the client 110 may query the server 120 for the information about the printers. The server 120 then returns the printer attributes and their corresponding values that describe the status, location, available features, etc.

[0013] Figure 2 shows an overview of the flow of the document data and attributes in a print management system that utilizes the extended print capabilities. The object identifier (OID) databases 111 and 125 store the object identifiers of all the attributes that the print management system supports. The attribute dictionaries (AD) 112 and 126 maintain the non-DPA syntax of any supported attributes. For every attribute that is not in DPA syntax, the complex syntax is broken down to components in DPA-syntax and stored in AD 112. Hence, given an attribute name or OID, the AD 112 returns a sequence of component names (or OID) and their DPA-syntax. The object databases (ODB) 127 and 129 that store the objects and their attribute-value pairs. The print data memory 128 is a temporary storage that keeps the actual contents of the document. The accounting log 130 is a data file that contains the accountable print job information.

[0014] When a user submits a print job by selecting documents, the printer and other attributes, the client 110 puts together this data and sends it to the server 120. The server 120 then reads the data and stores the document data in the print data memory 128 and creates a abstract entity, a job object, that contain the attributes specified in the data package from client 110 and stores it in the ODB 127. When the specified printer 160 is ready to print a new job, the server 120 retrieves the document data from the print data memory 128 and sends the document data to the specified printer 160. The server 120 also retrieves the job object attributes from the ODB 127, and based on these attributes, sends print controls to the specified printer 160. When the specified printer 160 completes printing of the document or printing is terminated for any other reason, (i.e. if it is canceled or aborted), the server 120 writes the accounting information to the accounting log 130.

[0015] Figure 3 shows in detail the flow of attributes and their corresponding values between the client 110 and the server 120. When the user specifies an attribute, the client 110 queries its OID database 111 with the textural name of the attribute. The object identifier (OID) database 112 returns a globally unique identifier, such as the one discussed above for "job-owner", i.e., 1.0.10175.1.3.1.3, and the syntax. If the syntax is non-standard, then the client 110 further looks in the attribute dictionary 112 for the standard syntax components broken down. The attribute value specified by the user is then interpreted according to this syntax and is grouped with the OID that identifies the attribute. The <attribute-OID, value> pair is encoded and sent by the client across to the server 120. The server 120 queries the OID database 125 and attribute dictionary 126 with the OID. Then based on the syntax, the server 120 puts the <attribute-OID, value> pair in the format that can be readily interpreted by the server 120. This is stored in the object database 127 as part of an object.

[0016] Figure 4 and 5 describe how the dynamic attributes can be introduced to the print management system 100. First, the dynamic attributes and syntaxes are registered in the server's OID database 125 and attribute dictionary 126. When a user queries the server 120 for the special attributes "xxx-supported attributes", where "xxx" is an object class, the dynamic attributes are then registered in the client's OID database 111 and attribute dictionary 112. From the users' point of view, there is nothing special about querying these attributes (i.e. querying the server is a standard function). However, the client performs the additional task of updating the databases 111 and 112 in the background.

[0017] Registering dynamic attributes to the server's database 125 and the attribute dictionary 126 can be achieved in two ways. One is by running a system administrator tool 200 that reads in a data file 164 that contains the attributes, OIDs and syntax. Another method is an automated registration which reads in the same data file (i.e., without shutting down the servers and using the system administration function, above). The system administrator tool 200 should be run manually when the system is down. The system administrator tool 200 registers the new attributes into the OID database 125. If the attribute syntax is non-standard, the system administrator tool 200 also registers the components of the attributes expressed in DPA-primitive types in the attribute dictionary 126.

[0018] The automated registration varies slightly depending on the associated object class of the dynamic attributes. Registering printer attributes will occur when new printers are added to the system. Registering other object attributes will occur when the system administrator sets the "extended-attribute-file". Typically, the file pointed to by this attribute contains job or document attributes for management of site-specific accounting information.

[0019] Figure 4 illustrates how dynamic printer attributes are introduced to the system 100. When a new

printer is added to the system, the system administrator creates an abstract printer object that is mapped to the physical device. In the request to create this abstract printer object, the system administrator may specify an "extended-printer-attribute-file" (XPAF file), which is the path to the file that contains the dynamic attribute names, the OIDs and the syntaxes. The server 120 reads this XPAF file, translates the information, and then stores the attribute-name and the OIDs in the OID database 125 and the non-standard syntaxes in the attribute dictionary 126, respectively. When the client 110 queries "printer-supported-attributes", the server 120 returns the names, the OIDs and the syntaxes of all printer dynamic attributes. The client 110 then updates its OID database 111 and the attribute dictionary 112.

[0020] Figure 5 illustrates how dynamic document or job attributes are introduced. As discussed before, a standard management request includes setting or modifying an attribute value of an object. When the system administrator sets "extended-attribute-file", the server 120 reads the file pointed to by this path, translates the information and updates its OID database 125 and the attribute dictionary 126. Updating the client databases 111 and 112 is the same as in updating the printer attributes, except in that the client 110 should query "job-supported-attributes" or "document-supported-attributes".

[0021] Figures 6A and 6B are a flowchart illustrating how document attribute information is passed through the system to the printerl60. Figure 6A shows the document flow on the clientside. Figure 6B shows the document flow on the server side.

[0022] Beginning at step S500, control continues to step S505, where the user specifies an attribute which has a corresponding attribute value. Next, at step S510 the client 110 queries its OID database 111 with the textural name of the attribute. At step S515, the OID database 111 returns the OID and the syntax to the client 110.

[0023] At step S520, the client 110 determines whether the syntax is standard (i.e., per DPA standards). If the syntax is non-standard, control continues to step S525. Otherwise, control jumps to step S540.

[0024] At step S525, the client 110 queries the attribute dictionary 112 for the standard-syntax components that form the non-standard-syntax attribute. Then, at step S530, the attribute dictionary 112 returns the components and syntax to the client 110. Next, at step S535, the attribute components are translated into internal representation and grouped into an <attribute-OID, attribute value> pair. The client 110 is then ready to process the next user-specified attribute. Control then jumps to step S540.

[0025] In step S540, the attribute OID and the attribute value are paired and translated into internal representations. Control then continues to step S545. At step S545, the client 110 determines if all attributes have been checked. If not, control returns to step S505. Otherwise, once all attributes with standard and non-standard syntax are processed, control continues to step S550.

[0026] At step S550, the client 110 encodes the <attribute OID, attribute value> pairs and sends them to the server 120.

[0027] In Fig. 6B, at step S555, the server 120 decodes the data from the client 110. Then, at step S560, the OID and the syntax are decoded. Next, at step S565, the server 120 determines if the syntax is standard. If the syntax is non-standard, control continues to step S570. Otherwise, control jumps to step S585.

[0028] At step S570, the server 120 queries its attribute dictionary 126 for the components broken down into standard syntax. Then, at step S575, the attribute dictionary 126 returns the components and the syntax to the server 120. Next, at step S580, the components are translated into internal representations and grouped into an attribute-value. Control then jumps to step S590. In contrast, at step S585, the attribute value is translated into an internal representation. Control then also continues to step S590

[0029] Thus, at step S590, the internal representation of the <attribute OID-attribute-value> pair is stored in the object database 129. Then, at step S595 when the document is ready to print, the supervisor 124 retrieves the document attributes from the object database 129 and sends control signals to the printer 160 according to the attribute settings. Control then continues to step S600, where the control routine stops

[0030] The following scenario illustrates how these dynamic OIDs function. Company X releases a new printer 162 that has the new finishing feature expressed in ASN.1 syntax, as shown in Fig. 7. This finishing feature is stored as part of the XPAF file 164. When the printer 162 is first connected to the system 100, the system reads the XPAF 164 file. The system 100 then translates the feature into a pseudo-C structure, as shown in Fig. 8.

[0031] A data structure denoting this complex syntax is stored in the attribute dictionary 114, and is retrievable at run-time. The client 110 can retrieve this attribute and use the information to build the globally unique identifier at run-time to allow the user to manipulate the dynamic attribute.

[0032] As shown in Figures 3-6B, as well as in the example outlined above, the implementation of dynamic OIDs is preferably performed on a programmed general purpose computer. However, the implementation of dynamic OIDs can also be performed on a special purpose computer, a programmed microprocessor or microcontroller and peripheral integrated circuit elements, an ASIC or other integrated circuit, a hardwired electronic or logic circuit such as a discrete element circuit, a programmable logic device such as a PLD, PLA, FGPA or PAL, or the like. In general, any device on which a finite state machine capable of implementing the flowchart shown in Figs. 6A and 6B and the example illustrated

above, can be used to perform the implementation of the dynamic OIDs.

Claims

1. A system for providing dynamic extension of print management capabilities, comprising:

   a first processor that receives an input signal containing dynamic attributes and sends the corresponding attributes to a second processor; and
   the second processor that receives the transmitted dynamic attributes from the first processor and performs actions based on the received dynamic attributes;
   wherein the first processor receives the dynamic attributes back from the second processor and translates the dynamic attributes into a user-recognizable form.

2. The system of claim 1, wherein each dynamic attribute has an object identifier, a syntax and a value.

3. The system of claim 1 or claim 2, wherein the first and second processors encode the OIDs.

4. A system according to any of claims 1 to 3, wherein the first processor further comprises:

   a first memory containing textual names of the dynamic attributes and the dynamic attributes' corresponding OIDs; and
   a second memory containing internal representations of the syntaxes of the dynamic attributes.

5. The system of claim 4, further comprising:

   a third memory containing textual names of the dynamic attributes and the corresponding OIDs; and
   a fourth memory containing internal representations of the syntaxes of the dynamic attributes; and
   a fifth memory containing the OIDs of the dynamic attributes and their values.

6. A method for providing dynamic extension of print management capabilities, comprising:

   receiving an input signal containing dynamic attributes at a first processor;
   sending the corresponding attributes to a second processor;
   performing actions based on the received dynamic attributes;
   returning the dynamic attributes back to the first processor; and
   translating the dynamic attributes into a user-recognizable form.

7. The method of claim 6, wherein each dynamic attribute received has an object identifier (OID), a syntax and a value.

8. The method of claim 6 or claim 7, further comprising encoding the OIDs.

9. The method of any of claims 6 to 8, further comprising storing textual names of the dynamic attributes, the dynamic attributes' corresponding OIDs, and internal representations of the syntaxes of the dynamic attributes.

10. The method of any of claims 6 to 8, further comprising storing textual names of the dynamic attributes and the corresponding OIDs, the internal representations of the syntaxes of the dynamic attributes, and the OIDs of the dynamic attributes and their values.

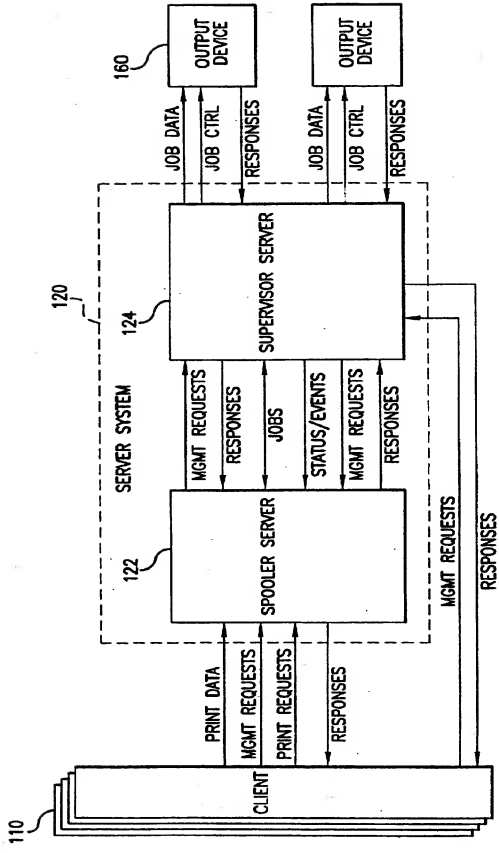

FIG.1

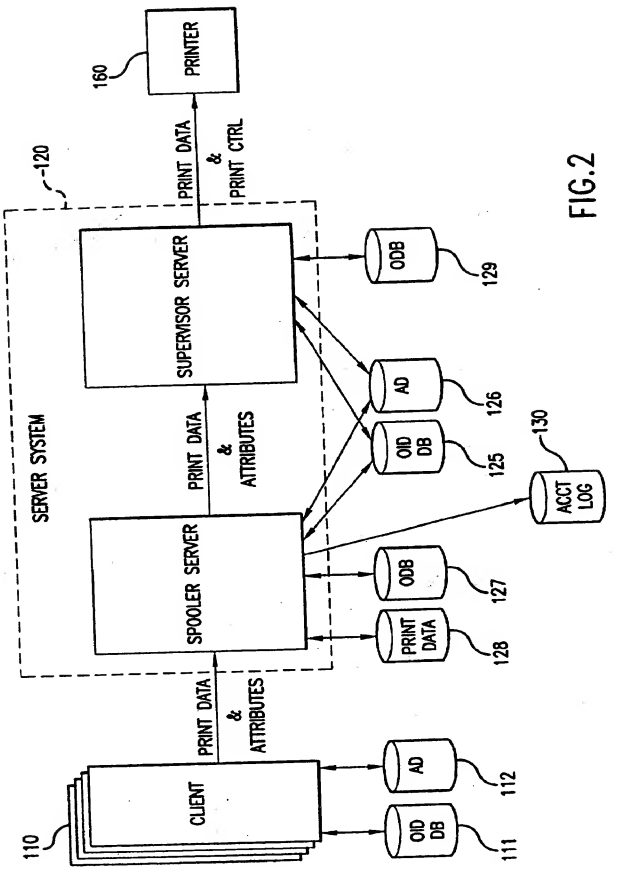

FIG.2

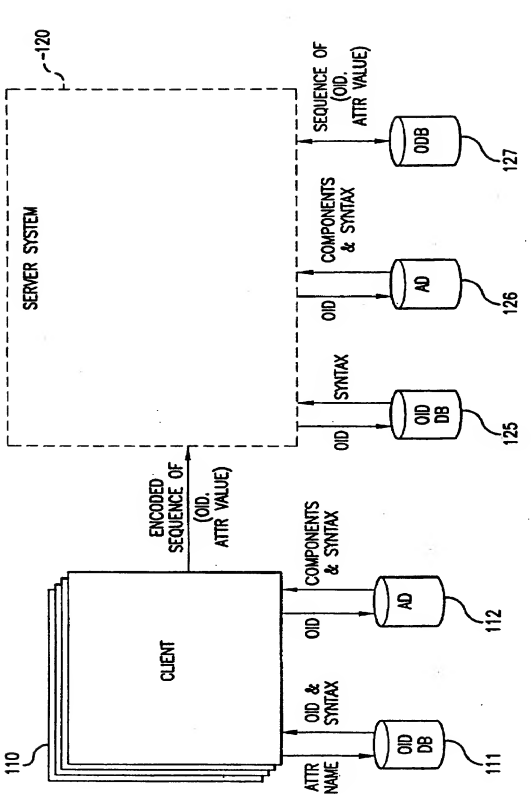

FIG.3

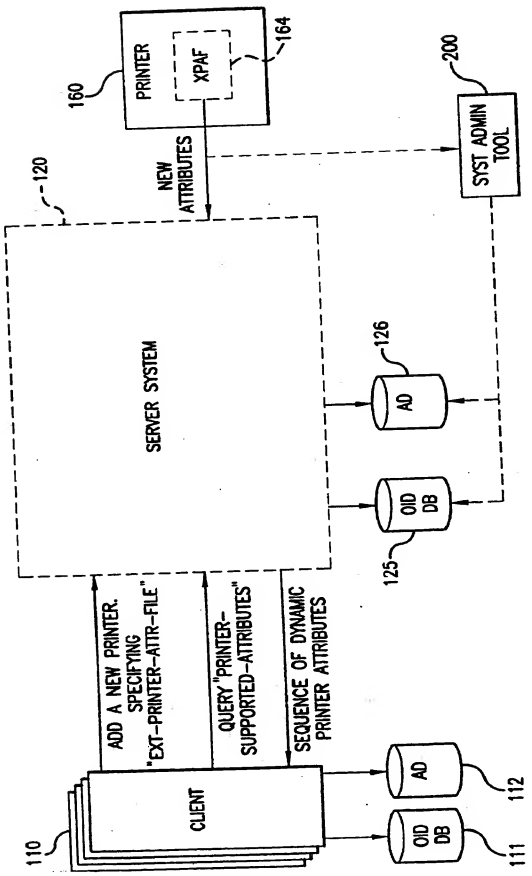

FIG.4

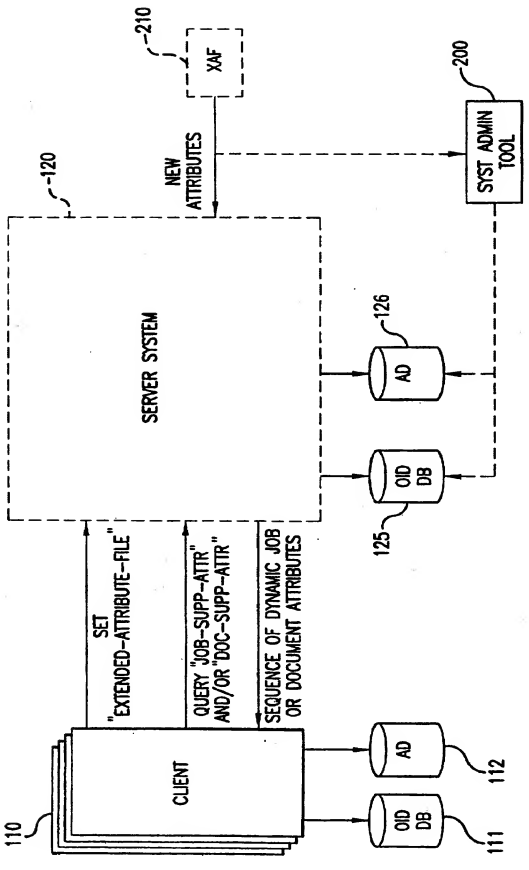

FIG.5

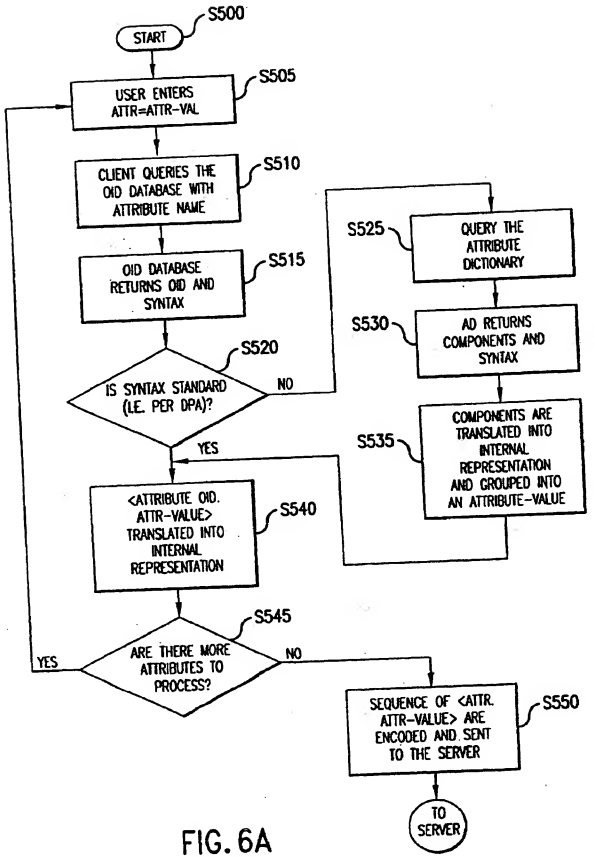

FIG. 6A

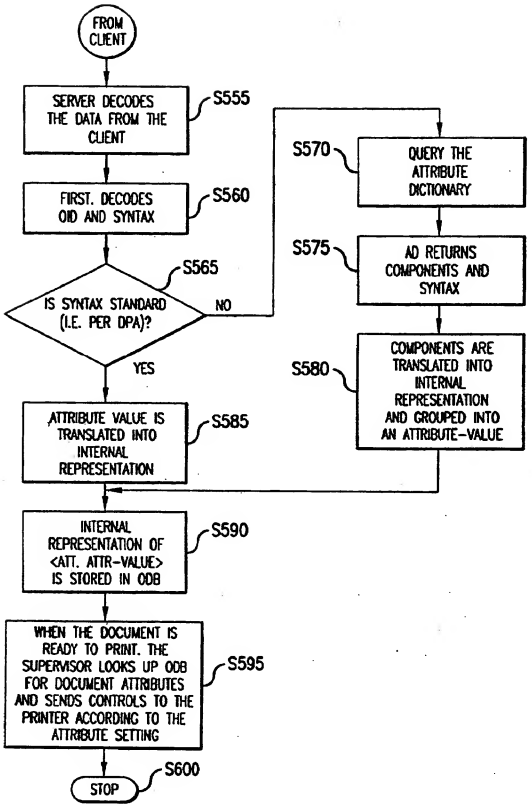

FIG.6B

```
Year2000Finishing::=ENUMERATED ( fin(0, ), fin1 (1) )
Year2000FinishingSyntax ATTRIBUTE-SYNTAX Year2000Finishing
     MATCHES FOR EQUALITY
     ::= { oc -run-time-syn 1 104 } // This defines the OID for this attribute.


BrandNewPrinterAttr  ::= CHOICE (
     year2000Finishing (0) Year2000Finishing
     year2000Features  (1) SEQUENCE {
          subFeature0      (0) VisibleString,
          subFeature1      (1) Intefer })


BrandNewPrinterAttrSyntax ATTRIBUTE-SYNTAX BrandNewPrinterAttr
     MATCHES FOR EQUALITY
     ::= ( oc-run-time 1 106)


oc-run-time being the OID root assigned to Xerox
```

FIG.7

```
year2000Finishing-value-0 = { 1.0.10175.3.5.5.1.105 };
     --assuming oc-run-time == 1.0.10175.3.5.5 )
brandnewPrinterAttr = (
     runtimeOID = { 1.0.10175.3.5.5.1.105 };
     aggregateType = AVT_RunTimeSequence: -- denotes level 0 syntax.
     -- components of this is an aggregate (sequence of) the following
     -- sub-components:


     (
          Intergeryear2000Finishing = fin0;
          year2000Feature is an aggregate with the following sub-sub-components:


     (
          aggregateType = AVT_RunTimeSequence
          subFeature0 = OCTET STRING ("Value for sub-feature zero");
          subFeature1 = 1234;
     }
}
```

FIG.8

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 917 043 A3

(12) EUROPEAN PATENT APPLICATION

(88) Date of publication A3:
12.04.2000 Bulletin 2000/15

(51) Int Cl.7: **G06F 3/12**

(43) Date of publication A2:
19.05.1999 Bulletin 1999/20

(21) Application number: **98309098.6**

(22) Date of filing: **06.11.1998**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **07.11.1997 US 966406**

(71) Applicant: **XEROX CORPORATION**
Rochester, New York 14644 (US)

(72) Inventors:
- **Yang, Jennifer Y.**
  Manhattan Beach, California 90266 (US)
- **Troung, Ton Huu**
  Westminster, California 92683 (US)
- **Nesbitt, David P.**
  Redondo Beach, California 90278 (US)

(74) Representative: **Skone James, Robert Edmund**
GILL JENNINGS & EVERY
Broadgate House
7 Eldon Street
London EC2M 7LH (GB)

(54) Dynamic extension of print capabilities

(57) A utility that defines additional attributes that would cater to a user's needs provides dynamically extended printing capabilities. The system architecture allows the information to be pushed down transparently to the receiving end, which understands the semantics of the given information. System administrators can define information to monitor for accounting purposes, etc. The utility allows additional printer features to be incorporated without disrupting the existing system.

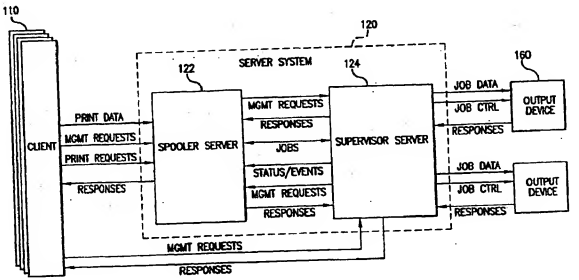

FIG.1

Printed by Jouve, 75001 PARIS (FR)

EP 0 917 043 A3

BNSDOCID: <EP_____0917043A3_I_>

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 98 30 9098

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.6)
X	EP 0 724 210 A (IBM) 31 July 1996 (1996-07-31) * figures 1,3,4 * * column 7, line 50 - column 9, line 30 * * column 4, line 18 - column 7, line 7 *	1,2,4-7, 9,10	G06F3/12
A	DE 44 22 619 A (HITACHI LTD) 5 January 1995 (1995-01-05) * figures 13-15,17,18 * * column 44, line 63 - column 53, line 64 *	1,2,4,5, 10	

TECHNICAL FIELDS SEARCHED (Int.Cl.6)

G06F
G06K

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	17 February 2000	Weiss, P

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 98 30 9098

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on 17-02-2000
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 0724210 A	31-07-1996	US 5659795 A	19-08-1997
DE 4422619 A	05-01-1995	JP 7072993 A	17-03-1995

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82